THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2009-11

Solution Techniques for Continuous Replenishment Inventory Routing Problems

Samuel Fomundam, Jeffrey Herrmann

Solution Techniques for Continuous Replenishment Inventory Routing Problems

Samuel Fomundam, Jeffrey W. Herrmann
Institute for Systems Research, University of Maryland, College Park, MD 20742

The Continuous Replenishment Inventory Routing Problem (CRIRP) is a special type of inventory routing problem (IRP) in which vehicle operations occur around the clock. The problem requires determining how many vehicles are needed to resupply the sites, which sites each vehicle should resupply, and the route that each vehicle should take. The objective is to minimize the number of vehicles. This technical report describes a special case of the CRIRP that is equivalent to the bin-packing problem. For the general problem, this report presents necessary and sufficient conditions for feasibility, a lower bound on the number of vehicles needed, and procedures for finding feasible solutions. These include solution construction heuristics and a genetic algorithm. We discuss the results of computational tests used to test the quality and computational effort of the heuristics. These results show that the route-building heuristic performs better than the other heuristics and the genetic algorithm.

1. Introduction

The Continuous Replenishment Inventory Routing Problem (CRIRP) is a special type of inventory routing problem (IRP) in which vehicle operations occur around the clock. Our study of the CRIRP is motivated by our work with public health officials who must plan the logistics for resupplying points of dispensing (PODs), which will dispense medications to the public in case of a public health emergency such as an anthrax attack. After receiving an initial but limited supply of medication, the PODs will operate continuously, around the clock, in order to give out thousands of doses of medication. Vehicles will resupply the PODs continuously from a central depot that has a stockpile of medication. Each vehicle repeatedly follows the same route, starting out as soon as it can after returning to the depot. At each site, the vehicle must deliver enough medication to satisfy demand until its next visit. Note that it is not necessary that all of PODs be resupplied at the same frequency. It may be more efficient for some PODs (especially those with high demand) to be resupplied more often than others.

A central concern is to determine how many vehicles are needed to resupply the sites, which sites each vehicle should resupply, and the route that each vehicle should take. Minimizing the number of vehicles is an important objective due to the limited number of available drivers and vehicles. Also, continuous replenishment means that the operating costs are related to the number of vehicles, which are continuously running.

Although motivated by this application, the CRIRP can occur in any setting where operations occur continuously and the resupply frequency of sites can vary. Problems with travel times of many days between deliveries can be viewed as continuous replenishment as well.

In a typical formulation of the IRP (see, for instance, Campbell *et al*., 1998), there is a single product that each customer consumes at a constant daily rate. Each customer also has a predetermined inventory capacity. A customer's existing inventory must not run out before a vehicle resupply. The IRP is solved over a planning horizon (for example, one week). There is a fleet of homogenous vehicles of a given capacity, and the objective is to minimize the cost of supplying the customers by identifying which days each customer should be supplied, determining the quantity to be supplied to each customer, and routing the fleet of vehicles to supply the determined quantities to the customers assigned to a particular day. Other notable work on the IRP includes Golden *et al*. (1984), Bard *et al*. (1998), and Jaillet *et al*. (2002). Moin and Salhi (2007) provide a recent review of the IRP.

In more recent work Campbell and Savelberg (2004a) take a two-phase approach to solving the IRP. The first phase uses integer programming to determine which customers to serve over the next several days and the quantities to be delivered. The results of the first phase are used as inputs for the second phase. This phase uses the VRP and scheduling techniques to plan delivery routes and schedules. Constraints encountered in the second phase may lead to a modification of the results obtained in the first phase. In another recent work, Campbell and Savelsbergh (2004b), present Vendor Managed Inventory Replenishment. In this version of the IRP, a vendor monitors customers' inventories and conducts replenishment of their inventories by coordinating inventory levels and vehicle deliveries to minimize long term costs.

In the IRP, vehicle routing decisions are made for each day. The routes start and end in the same day; they don't go into the next day. All of the vehicles are available at the beginning of the next day. There is a "jump" from one day to the next where no vehicles are operating. This characteristic does not exist in the CRIRP, in which customers are supplied continuously, around the clock. When vehicles return to the depot, they immediately reload and resupply their customers. These continuous operations are essential in a public health emergency.

Additionally, unlike the IRP, the CRIRP does not consider limits on the maximum inventory that can be stored at the customer sites. If they existed, such limits would place an upper bound on the duration of the routes that could be considered, a topic we address later in this paper.

The CRIRP problem can be viewed as a strategic IRP, in that we consider a fleet sizing problem that is similar to Webb and Larson (1995). However, unlike Webb and Larson, we consider the case where replenishments happen continuously. That is, a route begins as soon as the vehicle completes its route and returns to the depot.

This paper addresses the single-product, deterministic, steady-state problem in which the loading and unloading times at each site are modeled as a constant time. (This is reasonable if the marginal time needed to unload an item is small compared to the travel times.) Inventory is treated as a continuous variable. The storage capacity at each site is not given, for it will be set appropriately after the routing problem is solved. The depot always has enough inventory to load trucks.

The contribution of this paper is to introduce the CRIRP, which has not been previously studied, and to present a special case, a lower bound, heuristics, and a genetic algorithm. Fomundam (2008) developed a branch-and-bound scheme that can be used for small instances and tested a third, randomized heuristic that was shown to perform poorly. The conclusions discuss directions for future work on the problem.

2. Problem Formulation

In the CRIRP, there are n sites (customers). Each site (i = 1, …, n) has a demand rate of L_i items per time unit. This is the rate at which the site consumes material. There is a depot (i = 0) that has an unlimited amount of material. The time spent at site i (to refill a vehicle or deliver material) is p_i for i = 0, …, n. The time to travel from site i to site j is c_{ij}. The vehicles are identical, each with capacity of C items of material.

The problem is to find a feasible solution with the smallest number of vehicles. A feasible solution specifies a route for each vehicle, and each site is assigned to one route. The delivery amount at a site is the route duration multiplied by the site's demand rate.

A vehicle may visit the depot multiple times during a route to refill. A partial route that starts at the depot and ends at the depot is a "subroute." A vehicle may have multiple subroutes but visits each site just once on its route.

Given a solution, we evaluate its feasibility as follows. Let vehicle v have r subroutes. Let the sequence $s_{vj} = \{0, [1], ..., [k]\}$ be subroute j for vehicle v, where k is the number of sites on the subroute and [i] is the index of the i-th site visited. The total demand for the subroute is $D(s_{vj}) = L_{[1]} + \cdots + L_{[k]}$. The total time to complete the subroute is

$T\left(s_{vj}\right) = p_0 + c_{0[1]} + p_{[1]} + c_{[1][2]} + \cdots + p_{[k]} + c_{[k]0}$. The total time for vehicle *v* to complete all of its subroutes is $T_v = T\left(s_{v1}\right) + \cdots + T\left(s_{vr}\right)$.

When the vehicle visits site *i*, it will need to deliver $L_i T_v$ units of material in order to keep the site supplied until the vehicle's next visit. When vehicle *v* starts subroute s_{vj}, it should take $D\left(s_{vj}\right)T_v$ items in order to satisfy the demand of all the sites on that subroute; this quantity is the *load* of that subroute. Let $D_v^* = \max\left\{D\left(s_{v1}\right), \ldots, D\left(s_{vr}\right)\right\}$. The maximum load for vehicle *v* is $M_v = D_v^* T_v$. The solution is feasible if each site is assigned to exactly one vehicle and each vehicle's maximum load is not greater than the vehicle capacity. That is, $M_v \le C$ for all vehicles $v = 1, \ldots, K$.

In order to demonstrate the existence of feasible solutions, consider the trivial subroutes $z_i = \left\{0, i\right\}$, for $i = 1, \ldots, n$. Then, $T\left(z_i\right) = p_0 + c_{0i} + p_i + c_{i0}$ and $D\left(z_i\right) = L_i$. Clearly there are feasible solutions to CRIRP if and only if $D\left(z_i\right)T\left(z_i\right) \le C$ for all $i = 1, \ldots, n$.

The objective is to find a feasible solution with the minimal number of vehicles. It is easy to see that CRIRP is NP-hard, like virtually all vehicle routing problems (Lenstra and Rinnooy Kan, 1981).

3. Example

Consider the six-site problem instance (along with three subroutes) shown in Figure 1. At each site, the demand rate L_i (in items per time unit) is shown in parentheses. The service time $p_i = 1$ time unit at the depot and all sites. The travel time equals one time unit between the depot and sites 1, 2, 4, and 6 as well as between sites 2 and 3, between sites 3 and 4, between sites 5 and 6. The travel time between the depot and site 5 equals 1.4 time units.

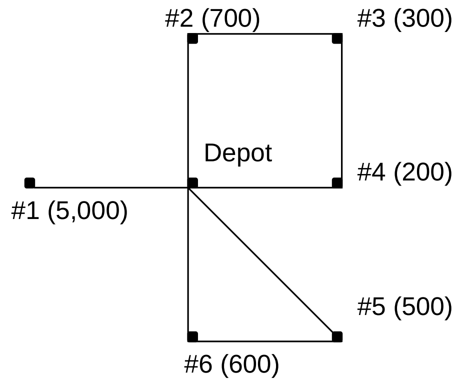

Figure 1. A six-site instance of the CRIRP, showing three subroutes.

If the vehicle capacity C = 20,000 items, then one feasible solution to the instance in Figure 1 has two vehicles. The first vehicle follows only one subroute $s_{11} = \{0,1\}$. The demand $D_1^* = D(s_{11}) = 5{,}000$ items per time unit, and the route duration $T_1 = T(s_{11}) = 4$ time units, so the load M_1 = 20,000 items. The second vehicle has two subroutes: $s_{21} = \{0,2,3,4\}$ and $s_{22} = \{0,5,6\}$. The first subroute demand $D(s_{21})$ = 1,200 items per time unit, and the subroute duration $T(s_{21}) = 8$ time units. The second subroute demand $D(s_{22})$ = 1,100 items per time unit, and the subroute duration $T(s_{22}) = 6.4$ time units. Therefore, the total route duration $T_2 = T(s_{21}) + T(s_{22}) = 14.4$ time units. $D_2^* = \max\{D(s_{21}), D(s_{21})\} = 1{,}200$ items, so $M_2 = D_2^* T_2 = 17{,}280$ items. The load for the first subroute equals 17,280 items, and the load for the second subroute equals 15,840 items.

4. The Special Case of Identical Demand

Consider the special case in which all $L_i = L$. (This special case is a useful model for the POD resupply problem if the jurisdiction's mass dispensing plans call for a set of identical PODs.) In this case, we can show that the non-trivial subroutes of a feasible solution can be split into the trivial subroutes without increasing the maximum load of any vehicle. Thus, there is an optimal solution in which every vehicle's route is the concatenation of trivial subroutes.

Consider a feasible solution in which a vehicle *v* visits *n* sites using *r* subroutes, where $r < n$. (The *n* here may be less than the number of sites in the entire problem.) Therefore, at least one subroute visits more than one site. Let $m_0 = 0$. Renumber the sites and define m_k ($k = 1,\ldots,r$) so that the first subroute visits sites $1,\ldots,m_1$, the second subroute visits sites $m_1+1,\ldots,m_2$, and so forth, with $m_r = n$.

Let $h = \max_{1\le k\le r}\left\{m_k - m_{k-1}\right\}$. Note that $h \ge 2$ and $hr \ge n$. Let TT_k be the travel time of subroute *k*. Note that $TT_k \ge c_{0i} + c_{i0}$ for any $i \in \left\{m_{k-1}+1,\ldots,m_k\right\}$.

Now consider the duration of each subroute *k*, and let T_0 be the duration of the current route:

$$T\left(s_{vk}\right) = p_0 + \sum_{i=m_{k-1}+1}^{m_k} p_i + TT_k$$

$$\begin{aligned} T_0 &= \sum_{k=1}^{r} T\left(s_{vk}\right) \\ &= rp_0 + \sum_{i=1}^{n} p_i + \sum_{k=1}^{r} TT_k \end{aligned}$$

On subroute *k* the demand $D\left(s_{vk}\right) = \left(m_k - m_{k-1}\right)L$. The maximum subroute demand is therefore *hL*, and the maximum load is hLT_0. Because the solution is feasible, $hLT_0 \le C$.

Now, modify this solution to construct a new solution in which this vehicle visits all of the same sites using trivial subroutes. Let $t_i = p_0 + c_{0i} + p_i + c_{i0}$ for all $i = 1,\ldots,n$. Let T_1 be the duration of the new route:

$$T_1 = \sum_{i=1}^{n} t_i = \sum_{i=1}^{n} \left(p_0 + c_{0i} + p_i + c_{i0} \right)$$
$$= np_0 + \sum_{i=1}^{n} p_i + \sum_{i=1}^{n} \left(c_{0i} + c_{i0} \right)$$

In this solution, the maximum subroute demand is L, and the maximum load is LT_1. Now, we will show that $LT_1 < hLT_0$ by proving that $hT_0 - T_1$ is positive.

$$hT_0 - T_1 = p_0 \left(hr - n \right) + \left(h - 1 \right) \sum_{i=1}^{n} p_i + \left(\sum_{k=1}^{r} hTT_k - \sum_{i=1}^{n} \left(c_{0i} + c_{i0} \right) \right)$$

Because $hr \geq n$, the first term is non-negative. Because $h \geq 2$, the second term is positive. To analyze the third term, we regroup the terms in the last summation by the subroutes to get the following:

$$\sum_{k=1}^{r} hTT_k - \sum_{i=1}^{n} \left(c_{0i} + c_{i0} \right) = \sum_{k=1}^{r} \left(hTT_k - \sum_{i=m_{k-1}+1}^{m_k} \left(c_{0i} + c_{i0} \right) \right)$$
$$\geq \sum_{k=1}^{r} \sum_{i=m_{k-1}+1}^{m_k} \left(TT_k - c_{0i} - c_{i0} \right)$$

Each term of this double summation is non-negative. Therefore, $hT_0 - T_1$ is positive, and $LT_1 < hLT_0 \leq C$. This shows that using the trivial subroutes is also feasible because they reduce the load of the vehicle. Therefore, there is an optimal solution with all trivial subroutes.

Which vehicle should do which subroutes? Let $t_i = p_0 + c_{0i} + p_i + c_{i0}$ for all $i = 1, \ldots, n$. Suppose vehicle v completes a set S_v of trivial subroutes. The route is feasible if and only if $M_v = L \sum_{i \in S_v} t_i \leq C$, which is equivalent to $\sum_{i \in S_v} t_i \leq C / L$. Thus, the problem becomes a bin packing problem in which the item size is t_i and the bin size is C/L. The packing of items into bins corresponds to the assignment of sites (and their trivial subroutes) to vehicles. Interestingly, the routing is trivial, because the load does not depend upon the sequence, so any sequence for a vehicle's route is sufficient in this special case. (Of course, the vehicle must follow the same sequence every time.)

5. Lower Bound

This special case can be used to justify the following lower bound for the more general case.

Given an instance *I* of CRIRP, let $L = \min\{L_i\}$. Modify the instance *I* by replacing each L_i by *L*. Any solution that is feasible for *I* is still feasible for the new instance because this change cannot increase the load of any subroute. In the new instance, the sites have identical demand, so we know that there is an optimal solution that uses all of the trivial subroutes. Let T_1 be the total duration of all of the trivial subroutes. Because the special case is essentially a bin-packing problem, we know that $\lceil T_1 L / C \rceil$ is a lower bound on the number of vehicles needed for the new instance and, consequently, a lower bound on the number of vehicles needed for the instance *I*.

We expect this bound to be tighter when all of the L_i are nearly equal but worse when the L_i have a large range.

5. Heuristics

Because CRIRP is NP-hard and we have no exact techniques that are useful for large instances, we developed and tested procedures for constructing solutions the problem. We know of no other existing techniques for this problem.

We generated five construction heuristics and developed a genetic algorithm. The following sections describe these procedures. The third, fourth, and fifth heuristics are route-first, cluster-second heuristics (cf. Beasley, 1983).

5.1 Bin-packing Heuristic

The first heuristic (which we denote *BP*) is a three-stage bin-packing approach that has a parameter W. The parameter W is varied from the greatest L_i to the sum of all L_i. In the first step, the heuristic uses the first-fit-decreasing algorithm to find a solution to the bin-packing problem in which each site i is an item, the item size is the demand rate L_i, and the bin capacity is W. This assigns sites to subroutes so that the subroute demands are roughly equal and no larger than W.

In the second stage, the heuristic uses the nearest neighbor algorithm on each subroute to find a path that begins and ends at the depot and visits all of the sites in that subroute. If $D(s_k)T(s_k) > C$ for any subroute k, then there is no feasible solution with this subroute. Any subroute k with $C / D(s_k) \geq T(s_k) > C/W$ is assigned to its own vehicle.

In the third stage, the heuristic uses the first-fit-decreasing algorithm to find a solution to the bin-packing problem in which each unassigned subroute k is an item, the item size is the subroute duration $T(s_k)$, and the bin capacity is C/W. This assigns these subroutes to vehicles. Each vehicle can visit its subroutes in any sequence. Each route has a duration no bigger than C/W, and each subroute has a demand that is no larger than W, so the route is feasible.

The BP heuristic loops over values of W from the maximal value of L_i to the sum of all L_i and keeps the best feasible solution found. In our implementation, this loop considered six values of W.

The computational effort of the BP heuristic depends upon N_W, the number of values of the parameter W that are considered. For a value of W, the bin packing heuristics in the first and third stages each take $\mathrm{O}(n\log n)$ effort, and the nearest neighbor algorithm requires $\mathrm{O}(n^2)$ effort, so the BP heuristic requires $\mathrm{O}(N_W n^2)$ effort.

5.2 Route-building Heuristic

The second heuristic (which we denote *RB*) builds routes by creating instances of the capacitated vehicle routing problem (CVRP), which has been studied extensively (see, for example, Toth and Vigo, 1998). First, the RB heuristic sequences the sites by demand rate from low to high. Then it begins building routes, starting with the low-demand sites. When a route has been built, the sites assigned to that route are removed from the instance, and the RB heuristic continues to build routes until all sites have been assigned to routes.

To build a route, the RB heuristic first finds a lower bound on how many of the unassigned sites can be placed feasibly into a route. Starting with the first unassigned site, it adds the trivial subroutes corresponding to those sites until adding the next one would cause the maximum vehicle load to exceed vehicle capacity. Let LL be the largest number of trivial subroutes that can fit into a feasible route. Then, starting with $N = LL + 1$, the RB heuristic tries to find a feasible route with the first N unassigned sites.

To find a feasible route, the RB heuristic constructs instances of CVRP that have the depot and the first N unassigned sites. The vehicle capacity remains equal to C, but the quantities to be delivered to the sites in the CVRP instance are determined by the duration bound B. The RB heuristic uses different values of B in the range from $C/L_{\max}$ down to $C/\sum L_i$ (where the max and the sum are taken over the set of N sites in the CVRP instance). For each value of B, the RB heuristic multiplies each site's demand rate by B to create that site's delivery quantity and then uses the Clarke-Wright savings algorithm to construct for a solution to the CVRP instance. If the total duration of the routes in the solution is less than B, then the solution forms a feasible route for the CRIRP problem by using the solution's routes as the subroutes for a vehicle. (If a feasible solution is found for one value of *B*, no more values of *B* need to be checked.)

If no feasible solution can be found for $N = LL + 1$, then the RB heuristic sets $N = LL$ and finds a feasible route (which must be possible because the *LL* trivial subroutes are a feasible solution.) Otherwise, it increases *N* by 1 and tries to find a feasible route for the expanded set of unassigned sites. It repeats this until, for some value of *N*, it can find no feasible route using the Clarke-Wright savings algorithm for any value of *B* (or all unassigned sites are in the feasible solution). At this point, the heuristic saves the last feasible solution found as a route for one vehicle (the routes from the CVRP solution become the subroutes for this vehicle). As mentioned above, the sites on that route are removed from the instance, and the RB heuristic continues with the remaining sites until all of them have been assigned to routes. The number of vehicles in the solution equals the number of feasible routes that were saved.

It is easy to see that a CRIRP route built from a feasible CVRP solution is feasible. Consider a subroute $s_{vj} = \{0,[1],...,[k]\}$. The total demand for the subroute is $D(s_{vj}) = L_{[1]} + \cdots + L_{[k]}$, and its load is therefore $D(s_{vj})T_v$, where T_v is the route duration. Because $T_v \leq B$, the load is not greater than $B(L_{[1]} + \cdots + L_{[k]}) = BL_{[1]} + \cdots + BL_{[k]}$, which is the sum of the delivery quantities for the sites on this subroute. Because these sites are a feasible route in the CVRP problem, this sum must be no greater than *C*. Therefore, the subroute load is not greater than *C*.

For example, if we consider the example from Section 3, then the RB heuristic will, at some point during its execution, consider the $N = 5$ smallest demand sites, which are sites 4, 3, 5, 6, and 2. When $B = C/L_{\max} = 20{,}000/700 = 28.57$, it creates an instance of CVRP with delivery quantities for sites 2 to 6 of 20,000, 8,570, 5,710, 14,290, and 17,140. A feasible solution to the CVRP has four routes: 0-4-5-0, 0-3-0, 0-6-0, and 0-2-0. The total duration is 19.24 time units, which is less than B, so this is a feasible CRIRP route. To confirm this, note that the maximum subroute demand is 700 items per time unit, so the maximum load is 13,470 items, which is less than the vehicle capacity.

However, when $N = 6$, the largest value of $B = 20{,}000/5{,}000 = 4$, and there is no feasible solution because it is impossible to find a feasible route with a duration less than or equal to 4. So the solution with five sites becomes the route for the first vehicle. A second vehicle is required for the last unassigned site (site 1).

In our implementation of the RB heuristic, the loop over B considered six equally-spaced values from $C/L_{\max}$ down to $C/\sum L_i$.

The computational effort of the RB heuristic depends upon N_B, the number of values of the duration bound B that are considered. The Clarke-Wright savings algorithm requires $\mathrm{O}\left(n^2 \log n\right)$ effort (Golden *et al.*, 1980), and this is performed up to N_B times in order to find a feasible route for N unassigned sites. Altogether, the RB heuristic will try to find at most n feasible routes. Thus, the RB heuristic requires $\mathrm{O}\left(N_B n^3 \log n\right)$ effort.

5.3 Space-filling curve heuristic

The third heuristic (which we denote *SFC*) builds routes in two steps. First, it generates a space-filling curve that begins at the depot and visits all of the sites. Then, it generates a feasible solution from this sequence.

To generate the space-filling curve, we use the procedure described in Bartholdi and Platzman (1988). The locations of the depot and sites are scaled and translated so that the depot is at the center of a unit square, and all of the sites fit into the unit square. The space-filling curve assigns each site to a position between 0 and 1. Because sites near 0 and 1 are in a corner of the unit square (and far away from the depot), we generate a sequence of sites by starting with the sites in the interval [7/8, 1] and then visiting the sites in the interval [0, 7/8).

For example, consider the depot and five sites shown in Figure D. The figure shows the scaled and translated position of each site and each site's position on the space-filling curve. Although A has the lowest position, the sequence starts with site E, which is in the interval [7/8, 1]. After site E, the sequence has sites A, B, C, and D in order by their position on the space-filling curve.

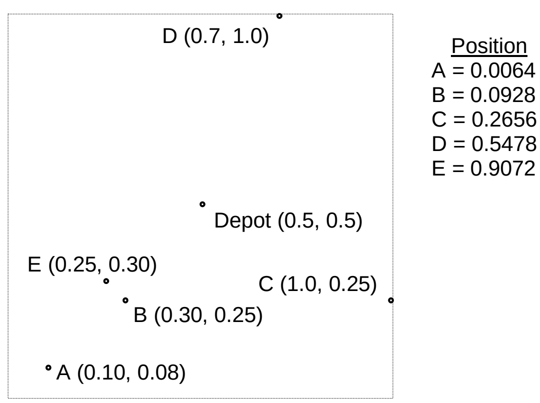

Figure D. Example with a depot and five sites, showing each site's position in the unit square and each site's position on the space-filling curve.

Given this sequence of sites, we then construct a feasible sequence by splitting it into routes and subroutes as follows.

We construct a graph with $n+1$ nodes, numbered from 1 to $n+1$. Nodes 1 to n represent the sites in the given sequence. That is, the node 1 is the first site in the sequence, node 2 is the second site in the sequence, and so forth. Node $n+1$ is a sink node.

Each node has four labels: $L_r(i)$ is the shortest length of a route starting at this node. $L_d(i)$ is the maximum subroute demand in the route starting at this node. $L_n(i)$ is the node at the end of the best arc. $L_f(i)$ denotes whether there is a feasible subroute starting at this node.

Set $L_r(n+1) = L_d(n+1) = L_n(n+1) = 0$.

There are arcs between nodes. Arc (i, j) represents a subroute that begins with the site at node i and ends with the site at node $j-1$. Associated with arc (i, j) are three values: $m(i, j)$ equals the duration of the subroute, including the trips from and to the depot and the load and unload times; $d(i, j)$ is the total subroute demand rate; and $f(i, j)$ denotes whether the subroute is feasible.

The basic idea of this routine is to build routes starting with the last site by finding shortest paths. Once one route is found, those sites are removed from the problem, and then it begins again with the last site not included.

S is the sequence of sites. $S(k)$ is the k-th site in the sequence.

```
Let L = n.  Set all node labels to zero.
repeat
    if L = 1 then create a route with only S(L)
    else
        L_r(L) = m(L,L+1), L_d(L) = d(L,L+1), L_n(L) = L+1, L_f(L) = 1.
        k = L – 1
        repeat
            for i = k+1 to L+1
                if (m(k,i) + L_r(i)) d(k,i) > C ) then f(k, i) = 0
                else
                    f(k, i) = 1
```

```
            if L_f(k) = 0 or m(k,i) + L_r(i) < L_r(k) then
                L_r(k) = m(k,i) + L_r(i), L_d(k) = max{d(k,i), L_d(i)}, L_n(k) = i,
                L_f(k) = 1
            endif
        endif
    end for
    if L_f(k) = 0 then
        set k = k+1
        create route starting at S(k)
    elseif L_r(k) L_d(k) ≤ C then
        if k = 1 then create route starting at S(k)
        else set k = k−1
        endif
    else
        B = L_r(k)
        repeat
            for i = k to L
                for j = i+1 to L+1
                    if B·d(k,i) > C then f(i, j) = 0
                    endif
                end for
            end for
            for i = L downto k
                L_f(i) = 0, L_r(i) = M
                for j = i+1 to L+1
                    if f(i, j) =1 and m(i, j) + L_r(j) < L_r(i) then
                        L_r(i) = m(i, j) + L_r(j), L_d(i) = max{d(i, j), L_d(j)},
                        L_n(i) = j, L_f(i) = 1
                    endif
                end for
            end for
            if L_f(k) = 0 then
                set k = k+1
                create route starting at S(k)
            elseif L_r(k) L_d(k) ≤ C then
                if k = 1 then create route starting at S(k)
                else we have a feasible route; set k = k−1
                endif
            else B = L_r(k)
            endif
```

```
                until we find a feasible route or created a route
            endif
        until we created a route
        set L = k − 1
    endif
until all sites are assigned to routes
```

For example, given the six-site example presented earlier, consider the sequence (1, 2, 3, 4, 5, 6). The algorithm first finds a feasible route with one subroute containing sites 5 and 6. Then it finds a feasible route with one subroute containing sites 4, 5, and 6. Then it finds a feasible route with one subroute containing sites 3, 4, 5, and 6.

When it gets to k = 2, it first determines that there is a shortest path with two subroutes: 2 and 3-4-5-6. The duration is approximately 14.4 time units. But the demand rate of the second subroute is 1600, so the load is over 23,000, which exceed the vehicle capacity. Thus, the algorithm eliminates the arcs corresponding to subroutes that have a demand rate greater than 20000/14.4 = 1389 and finds a shortest path using the remaining arcs. This creates a feasible route with two subroutes: 2-3 and 4-5-6. When k = 1, there are no feasible subroutes, so k = 2, and the previous feasible route is created. The only remaining site is site 1, so the algorithm creates a second route with only this site.

Note that this algorithm assumes that all trivial subroutes are feasible and is thus guaranteed to find a feasible solution. It is easy to see that, given a sequence that corresponds to an optimal solution, this algorithm will generate a solution with the same number of routes (the composition of the routes and subroutes could be different, however).

The computational effort of sequencing the sites using the space-filling is $O(n)$ to generate the positions and $O(n^2)$ to sort the sites based on position.

The computational effort of the splitting algorithm is $O(n^4)$. The loop to find a new shortest path if the current one is infeasible eliminates at least one arc on each pass; moreover, these arcs, once eliminated, never become feasible again. Therefore, the loop can be run at most $O(n^2)$ times over the course of the algorithm. The loop itself requires $O(n^2)$ time to find the shortest path.

5.4 Sweep heuristic

The fourth heuristic (which we denote *SWP*) is also a route-first-cluster-second heuristic. First, it sequences the sites using a simplified version of the sweep algorithm (Gillett and Miller, 1974).

The algorithm translates all of the sites so that the depot is at (0, 0), determines each site's location in polar coordinates (with vectorial angles between -180 and 180 degrees) for each translated site, and sorts the sites by their vectorial angles to generate a sequence.

Given this sequence of sites, we then construct a feasible sequence by splitting it into routes and subroutes using the splitting algorithm described above.

The computational effort of sequencing the sites is $O(n)$ to generate the vectorial angles and $O(n^2)$ to sort the sites based on position. The computational effort of the splitting algorithm is $O(n^4)$, as described above.

5.5 Nearest neighbor heuristic

The fifth heuristic (which we denote *NN*) is also a route-first-cluster-second heuristic. First, it generates a tour (starting at the depot) using the standard nearest neighbor algorithm.

Given this sequence of sites, we then construct a feasible sequence by splitting it into routes and subroutes using the splitting algorithm described above.

The computational effort of sequencing the sites is $\mathrm{O}\left(n^2\right)$. The computational effort of the splitting algorithm is $\mathrm{O}\left(n^4\right)$, as described above.

5.6 Genetic algorithm

In addition to the above heuristics, we implemented a hybrid genetic algorithm (GA) based on the one that Prins (2004) developed to solve the VRP. Like the Prins GA, the chromosome is simply a sequence of sites. We use the splitting procedure described above to construct a feasible CRIRP solution from a sequence. We use the same order crossover as Prins. We tested versions using a simple mutation operator and using a local search as a mutation operator. In both the mutation and the local search, we used the nine types of moves described by Prins and added a tenth move that combines two routes.

Like Prins, we avoid clones in the population to avoid premature convergence and enforce a minimum difference in solution quality. We do allow solutions with the same number of routes (vehicles) if the sum of the route durations is different by a constant DELTA. If two solutions have the same number of routes, then the one with the smaller sum of the route durations has a better fitness.

Like Prins, we select parents with the binary tournament method. If the child is mutated and the mutation is not well spaced, then we try to add the child instead. The GA is controlled by two parameters, α, an upper bound on the number of replacements, and β, an upper bound on the number of replacements without improving the best solution.

To initialize the GA, we use the SFC, SWP, and NN algorithms described above to generate initial solutions. The remaining individuals are generated randomly. No local search is used to improve these solutions.

We set $\sigma = 30$ and set δ equal to the total route duration of the SFC solution divided by 1000. We set the probability of mutation to 0.1, as a high mutation rate is consistent with this type of GA. (Prins uses mutation rates of 0.05, 0.1, and 0.2.) We ran five trials of the GA on each instance.

We conducted a pilot study to see how different initial solutions, using local search or simple mutation, and run length affected the GA performance. The results are discussed in the next section.

6. Computational Tests

The purpose of the computational tests was to evaluate the relative performance of the heuristics. The heuristics were coded in Matlab. We used an implementation of the Clarke-Wright saving algorithm from Matlog, the Logistics Engineering Matlab Toolbox, created by Michael G. Kay at North Carolina State University.

To test the heuristics developed, we use four sets of location data obtained from the TSPLIB, a library of sample instances that provide either location data or the costs associated with the paths of a graph. They serve as test data for TSP solvers. We selected the following 4 sets of data:

- Burma 14: 14 cities in Burma; and
- Ulyssess 22: 22 locations from the Odyssey of Ulysses.
- Berlin 52: 52 locations in Berlin, Germany;
- Bier 127: 127 beer gardens in Augsburg, Germany;

In each of these four data sets, the locations are sequentially indexed using positive integers. Each location also has Cartesian coordinates. Although the data are sufficient for testing TSP solvers, more data is needed for the CRIRP.

We made the first location the depot. The other locations are then designated as sites and numbered from 1. We used the Euclidean distance between each pair of sites as the (symmetric) travel times between the sites. We then calculated the average travel time A of the instance. We then specified four values for the load time: $A/50$, $A/5$, A, and $3A$. (Every site in the data set had the same load time.)

We constructed three values for the vehicle capacity for each data set. To do this, we set the load times equal to the $3A$ and then found the maximum of the durations of the trivial subroutes for those sites. We multiplied this longest duration by 400 to get the maximum subroute load. (As discussed below, 400 is an upper bound on site demand rate.) We multiplied this by 1.5, 5, and 10 to get the values for the vehicle capacity.

We arbitrarily chose an average demand rate of 200 items per time unit. The depot demand rate was set to zero. We then generated a set of samples from a standard normal distribution. (Samples less than -2.5 and samples greater than 2.5 were discarded.) We constructed three sets of demand rates using three different values for the standard deviation: 80, 40, and 20. The demand rates at each site were determined by multiplying the standard deviation by the sample and adding 200 (so the average demand rate was approximately 200). Note that all of the demand rates are between 0 and 400.

This scheme ensures that the trivial subroutes are all feasible routes. Therefore each instance has at least one feasible solution.

In this manner, for each of the four TSP data sets, we created 4 load times, 3 sets of demand rates, and 3 vehicle capacities. Thus, we generated 36 instances of CRIRP for each TSP data set, giving us a total of 144 instances.

We determined the lower bound and ran the heuristics on each CRIRP instance.

We also tracked the time needed to run the heuristics. The average time for the BP heuristic was 0.21 seconds, but the time ranged from 0.02 seconds on the 14-site instances to 0.66 seconds on the 127-site instances. The average time for the RB heuristic was 0.94 seconds, but the time ranged from 0.06 seconds on the 14-site instances to 3.05 seconds on the 127-site instances. The computational effort of the BP heuristic was not significantly affected by other changes in the instance, but the computational effort of the RB heuristic decreased when the vehicle capacity was small or when the load time was high. This occurs because, in these scenarios, long routes are not feasible, so the RB heuristic needed less time to build a route.

Tables 1 and 2 are summaries of the results, aggregated over different problem parameters. The number in parentheses is the number of instances in that category.

The RB heuristic always found the best solution of those generated by the construction heuristics. For 58 instances, the number of vehicles equaled the lower bound. The BP heuristic found equally good solutions in 102 instances. These included 20 instances with small capacity, 38 with medium capacity, and 44 with large capacity. The performance of the SFC, SWP, and NN heuristics was similar.

Not surprisingly, as vehicle capacity increases, the number of vehicles needed reduces. The performance of the heuristics are nearly identical when the vehicle capacity is high. When the vehicle capacity is low, the solutions that the RB heuristic generates are significantly better, while the solutions that the BP heuristic generates are worse.

We tested different versions of the genetic algorithm. First, we considered different initial sequences. We tried four options: including the SFC solution, including the SWP solution, including the NN solution, and including all three solutions. We ran five replications of this GA using the simple mutation and with short runs (α_{max} = 300 and β_{max} = 100). The results showed that there was no significant difference in solution quality or run time.

We also considered using local search instead of a simple mutation. We ran five replications of the GA on the Berlin 52 instances using the local search and with short runs (α_{max} = 300 and β_{max} = 100). The results showed that using local search increased run times dramatically but did not lead to better solutions.

We also ran much longer replications. We ran five replications of the GA on the Berlin 52 instances using simple mutation and with long runs (α_{max} = 30000 and β_{max} = 10000). The results showed that using the longer runs increased run times dramatically but did not lead to better solutions.

Table 1. CRIRP problem instances and heuristic performance.

	Average number of vehicles					
Instances	LB	BP	RB	SFC	SWP	NN
Burma 14	1.17	1.61	1.50	1.61	1.61	1.58
Ulysses 22	1.25	1.86	1.78	1.86	1.89	1.83
Berlin 52	2.22	4.92	4.11	4.58	4.53	4.50
Bier 127	3.39	10.31	7.94	9.44	9.50	9.28
Low capacity	3.33	9.23	7.17	8.40	8.44	8.29
Medium capacity	1.52	2.98	2.63	2.92	2.90	2.83
High capacity	1.17	1.81	1.71	1.81	1.81	1.77

Table 2A. Number of vehicles in solutions generated by different heuristics for the Burma 14 instances.

Instance Number	Heuristics					
	Lower Bound	BP	RB	SFC	SWP	NN
1	1	5	4	5	5	5
2	1	3	2	3	3	3
3	1	2	2	2	2	2
4	1	2	1	1	1	1
5	3	5	4	5	5	5
6	2	2	2	2	2	2
7	1	2	2	2	2	2
8	1	1	1	1	1	1
9	3	4	4	5	5	4
10	2	2	2	2	2	2
11	1	2	2	2	2	2
12	1	1	1	1	1	1
13	1	2	2	2	2	2
14	1	1	1	1	1	1
15	1	1	1	1	1	1
16	1	1	1	1	1	1
17	1	2	2	2	2	2
18	1	1	1	1	1	1
19	1	1	1	1	1	1
20	1	1	1	1	1	1
21	1	2	2	2	2	2
22	1	1	1	1	1	1
23	1	1	1	1	1	1
24	1	1	1	1	1	1
25	1	1	1	1	1	1
26	1	1	1	1	1	1
27	1	1	1	1	1	1
28	1	1	1	1	1	1
29	1	1	1	1	1	1
30	1	1	1	1	1	1
31	1	1	1	1	1	1
32	1	1	1	1	1	1
33	1	1	1	1	1	1
34	1	1	1	1	1	1
35	1	1	1	1	1	1
36	1	1	1	1	1	1

Table 2B. Number of vehicles in solutions generated by different heuristics for the Ulysses 22 instances.

Instance Number	Heuristics					
	Lower Bound	BP	RB	SFC	SWP	NN
1	2	7	5	7	7	6
2	1	3	3	3	3	3
3	1	2	2	2	2	2
4	1	2	2	2	2	2
5	3	6	6	6	7	6
6	2	3	3	3	3	3
7	1	2	2	2	2	2
8	1	2	2	2	2	2
9	4	6	5	6	6	6
10	2	3	3	3	3	3
11	1	2	2	2	2	2
12	1	2	2	2	2	2
13	1	2	2	2	2	2
14	1	1	1	1	1	1
15	1	1	1	1	1	1
16	1	1	1	1	1	1
17	1	2	2	2	2	2
18	1	1	1	1	1	1
19	1	1	1	1	1	1
20	1	1	1	1	1	1
21	2	2	2	2	2	2
22	1	1	1	1	1	1
23	1	1	1	1	1	1
24	1	1	1	1	1	1
25	1	1	1	1	1	1
26	1	1	1	1	1	1
27	1	1	1	1	1	1
28	1	1	1	1	1	1
29	1	1	1	1	1	1
30	1	1	1	1	1	1
31	1	1	1	1	1	1
32	1	1	1	1	1	1
33	1	1	1	1	1	1
34	1	1	1	1	1	1
35	1	1	1	1	1	1
36	1	1	1	1	1	1

Table 2C. Number of vehicles in solutions generated by different heuristics for the Berlin 52 instances.

Instance Number	Heuristics					
	Lower Bound	BP	RB	SFC	SWP	NN
1	4	25	17	21	21	20
2	2	10	8	9	10	9
3	1	6	4	5	5	5
4	1	5	4	4	4	4
5	8	20	16	18	17	18
6	4	9	7	8	8	8
7	2	5	4	5	5	5
8	2	4	4	4	4	4
9	11	18	15	16	16	16
10	5	8	7	8	7	8
11	3	4	4	4	4	4
12	3	4	3	4	4	4
13	2	7	5	6	6	6
14	1	3	3	3	3	3
15	1	2	2	2	2	2
16	1	2	2	2	2	2
17	3	6	5	6	5	5
18	2	3	3	3	3	3
19	1	2	2	2	2	2
20	1	2	2	2	2	2
21	4	5	5	5	5	5
22	2	3	2	3	3	3
23	1	2	2	2	2	2
24	1	1	1	1	1	1
25	1	3	3	4	4	3
26	1	2	2	2	2	2
27	1	1	1	1	1	1
28	1	1	1	1	1	1
29	2	3	3	3	3	3
30	1	2	2	2	2	2
31	1	1	1	1	1	1
32	1	1	1	1	1	1
33	2	3	3	3	3	3
34	1	2	2	2	2	2
35	1	1	1	1	1	1
36	1	1	1	1	1	1

Table 2D. Number of vehicles in solutions generated by different heuristics for the Bier 127 instances.

Instance Number	Heuristics					
	Lower Bound	BP	RB	SFC	SWP	NN
1	1	57	34	44	43	43
2	1	22	15	20	20	20
3	1	12	8	10	11	10
4	1	10	7	8	9	8
5	15	44	33	40	39	39
6	7	19	15	18	18	18
7	4	10	8	10	10	10
8	3	8	6	8	8	8
9	22	39	33	36	37	36
10	10	17	14	16	17	16
11	6	9	8	9	9	9
12	5	7	6	7	7	7
13	1	14	10	14	14	13
14	1	7	5	7	7	6
15	1	4	3	4	4	3
16	1	3	3	3	3	3
17	5	13	10	12	12	12
18	2	6	5	6	6	6
19	2	3	3	3	3	3
20	1	3	3	3	3	3
21	7	11	10	11	11	11
22	3	5	5	5	5	5
23	2	3	3	3	3	3
24	2	3	2	2	2	2
25	1	7	6	7	7	7
26	1	4	3	4	4	3
27	1	2	2	2	2	2
28	1	2	2	2	2	2
29	3	7	5	6	6	6
30	1	3	3	3	3	3
31	1	2	2	2	2	2
32	1	2	2	2	2	2
33	4	6	5	6	6	6
34	2	3	3	3	3	3
35	1	2	2	2	2	2
36	1	2	2	2	2	2

Table 3A. Heuristic run times for the Burma 14 instances.
(All times in seconds.)

Instance Number	Heuristics BP	RB	SFC	SWP	NN
1	0.2463	1.0038	0.3017	0.0352	0.0059
2	0.0163	0.0511	0.0049	0.0018	0.0023
3	0.0161	0.0646	0.0034	0.0021	0.0025
4	0.0152	0.0312	0.0038	0.0026	0.0027
5	0.0145	0.0509	0.0030	0.0017	0.0020
6	0.0163	0.0522	0.0031	0.0019	0.0021
7	0.0161	0.0794	0.0036	0.0024	0.0023
8	0.0158	0.0211	0.0038	0.0025	0.0028
9	0.0146	0.0524	0.0030	0.0017	0.0019
10	0.0163	0.0536	0.0029	0.0019	0.0021
11	0.0161	0.1020	0.0039	0.0026	0.0029
12	0.0159	0.0205	0.0037	0.0025	0.0027
13	0.0151	0.0769	0.0034	0.0022	0.0024
14	0.0262	0.0170	0.0037	0.0025	0.0027
15	0.0183	0.0172	0.0037	0.0025	0.0027
16	0.0185	0.0169	0.0039	0.0025	0.0027
17	0.0160	0.0790	0.0034	0.0023	0.0024
18	0.0184	0.0203	0.0037	0.0025	0.0028
19	0.0191	0.0203	0.0037	0.0025	0.0027
20	0.0196	0.0204	0.0037	0.0025	0.0027
21	0.0161	0.0842	0.0041	0.0023	0.0026
22	0.0184	0.0203	0.0045	0.0025	0.0027
23	0.0191	0.0204	0.0037	0.0025	0.0027
24	0.0194	0.0205	0.0037	0.0025	0.0027
25	0.0164	0.0170	0.0037	0.0025	0.0027
26	0.0186	0.0169	0.0037	0.0025	0.0027
27	0.0185	0.0169	0.0037	0.0025	0.0026
28	0.0183	0.0170	0.0037	0.0025	0.0027
29	0.0174	0.0203	0.0038	0.0026	0.0028
30	0.0192	0.0203	0.0037	0.0025	0.0028
31	0.0192	0.0204	0.0038	0.0025	0.0027
32	0.0194	0.0204	0.0037	0.0025	0.0027
33	0.0175	0.0203	0.0038	0.0025	0.0027
34	0.0192	0.0203	0.0037	0.0025	0.0027
35	0.0193	0.0205	0.0037	0.0025	0.0027
36	0.0194	0.0204	0.0037	0.0025	0.0027

Table 3B. Heuristic run times for the Ulysses 22 instances.
(All times in seconds.)

Instance Number	Heuristics				
	BP	RB	SFC	SWP	NN
1	0.0257	0.0866	0.0046	0.0028	0.0031
2	0.0256	0.1268	0.0052	0.0032	0.0035
3	0.0281	0.1535	0.0067	0.0045	0.0040
4	0.0282	0.1713	0.0073	0.0051	0.0042
5	0.0275	0.0932	0.0047	0.0028	0.0030
6	0.0271	0.1318	0.0053	0.0033	0.0034
7	0.0286	0.1557	0.0067	0.0047	0.0040
8	0.0301	0.1654	0.0070	0.0051	0.0042
9	0.0275	0.0874	0.0048	0.0028	0.0031
10	0.0273	0.1345	0.0053	0.0033	0.0060
11	0.0286	0.1692	0.0069	0.0047	0.0041
12	0.0300	0.2436	0.0073	0.0058	0.0047
13	0.0269	0.1196	0.0057	0.0038	0.0040
14	0.0304	0.0390	0.0076	0.0058	0.0061
15	0.0335	0.0390	0.0074	0.0055	0.0059
16	0.0335	0.0390	0.0076	0.0056	0.0060
17	0.0295	0.1429	0.0058	0.0038	0.0040
18	0.0316	0.0516	0.0077	0.0057	0.0060
19	0.0344	0.0518	0.0075	0.0056	0.0060
20	0.0345	0.0515	0.0074	0.0056	0.0059
21	0.0302	0.1516	0.0059	0.0040	0.0042
22	0.0314	0.0517	0.0076	0.0057	0.0060
23	0.0342	0.0519	0.0075	0.0056	0.0059
24	0.0342	0.0518	0.0074	0.0055	0.0060
25	0.0305	0.0391	0.0078	0.0058	0.0061
26	0.0334	0.0388	0.0075	0.0056	0.0059
27	0.0347	0.0388	0.0074	0.0055	0.0058
28	0.0350	0.0389	0.0077	0.0055	0.0059
29	0.0300	0.0520	0.0077	0.0059	0.0061
30	0.0346	0.0518	0.0075	0.0056	0.0060
31	0.0355	0.0518	0.0074	0.0055	0.0058
32	0.0357	0.0517	0.0075	0.0056	0.0059
33	0.0298	0.0520	0.0077	0.0057	0.0061
34	0.0343	0.0517	0.0076	0.0056	0.0061
35	0.0352	0.0517	0.0075	0.0055	0.0059
36	0.0356	0.0518	0.0074	0.0055	0.0059

Table 3C. Heuristic run times for the Berlin 52 instances.
(All times in seconds.)

Instance Number	Heuristics				
	BP	RB	SFC	SWP	NN
1	0.1130	0.2257	0.0119	0.0074	0.0083
2	0.1102	0.2857	0.0117	0.0071	0.0082
3	0.1100	0.4419	0.0140	0.0091	0.0108
4	0.1099	0.5257	0.0157	0.0107	0.0119
5	0.1225	0.2280	0.0115	0.0069	0.0079
6	0.1197	0.2996	0.0119	0.0075	0.0085
7	0.1189	0.4480	0.0157	0.0096	0.0109
8	0.1188	0.5431	0.0155	0.0112	0.0120
9	0.1250	0.2326	0.0112	0.0068	0.0082
10	0.1230	0.3032	0.0130	0.0077	0.0087
11	0.1224	0.4816	0.0150	0.0106	0.0118
12	0.1223	0.5145	0.0166	0.0127	0.0134
13	0.1099	0.4023	0.0127	0.0082	0.0091
14	0.1096	0.6645	0.0173	0.0128	0.0139
15	0.1091	0.9605	0.0245	0.0191	0.0183
16	0.1198	1.1100	0.0323	0.0271	0.0249
17	0.1216	0.4047	0.0131	0.0088	0.0098
18	0.1189	0.7095	0.0177	0.0134	0.0157
19	0.1188	1.0070	0.0245	0.0207	0.0195
20	0.1261	1.4527	0.0327	0.0284	0.0250
21	0.1231	0.4134	0.0135	0.0091	0.0101
22	0.1225	0.5329	0.0207	0.0160	0.0164
23	0.1220	1.2503	0.0299	0.0252	0.0235
24	0.1317	0.3035	0.0354	0.0299	0.0305
25	0.1095	0.5948	0.0168	0.0115	0.0134
26	0.1116	1.1893	0.0243	0.0197	0.0206
27	0.1198	0.2210	0.0355	0.0305	0.0308
28	0.1285	0.2187	0.0350	0.0297	0.0302
29	0.1193	0.6729	0.0176	0.0127	0.0146
30	0.1256	1.1246	0.0272	0.0233	0.0243
31	0.1296	0.2969	0.0352	0.0305	0.0312
32	0.1394	0.2967	0.0349	0.0298	0.0300
33	0.1226	0.7362	0.0185	0.0142	0.0149
34	0.1221	1.2893	0.0329	0.0281	0.0280
35	0.1313	0.3041	0.0353	0.0306	0.0309
36	0.1418	0.3040	0.0350	0.0299	0.0303

Table 3D. Heuristic run times for the Bier 127 instances.
(All times in seconds.)

Instance Number	Heuristics				
	BP	RB	SFC	SWP	NN
1	0.6049	0.6047	0.0323	0.0214	0.0237
2	0.5954	0.8566	0.0309	0.0198	0.0228
3	0.5910	1.3584	0.0388	0.0249	0.0326
4	0.5923	1.6940	0.0425	0.0285	0.0383
5	0.6917	0.5808	0.0311	0.0204	0.0229
6	0.6802	0.8434	0.0307	0.0199	0.0233
7	0.6789	1.3604	0.0387	0.0267	0.0323
8	0.6754	1.6123	0.0428	0.0308	0.0408
9	0.7203	0.5886	0.0302	0.0200	0.0222
10	0.7132	0.8203	0.0314	0.0203	0.0238
11	0.7089	1.3849	0.0414	0.0281	0.0342
12	0.7092	1.5544	0.0480	0.0329	0.0419
13	0.5920	1.1583	0.0333	0.0224	0.0250
14	0.5917	2.0215	0.0477	0.0363	0.0403
15	0.5879	3.2075	0.0728	0.0592	0.0754
16	0.5887	4.4894	0.0956	0.0695	0.0805
17	0.6764	1.1445	0.0342	0.0238	0.0266
18	0.6745	2.1219	0.0503	0.0385	0.0416
19	0.6720	3.3967	0.0778	0.0657	0.0753
20	0.6715	4.0983	0.0954	0.0766	0.0858
21	0.7086	1.1626	0.0356	0.0247	0.0275
22	0.7086	2.2547	0.0528	0.0417	0.0456
23	0.7065	3.8589	0.0793	0.0680	0.0689
24	0.7056	3.5016	0.1084	0.0922	0.1142
25	0.5911	1.9364	0.0442	0.0336	0.0360
26	0.5902	3.3200	0.0761	0.0640	0.0721
27	0.5887	6.9229	0.1226	0.1090	0.0969
28	0.5929	6.8051	0.1713	0.1311	0.1312
29	0.6765	1.9185	0.0505	0.0377	0.0408
30	0.6771	3.6014	0.0780	0.0679	0.0697
31	0.6763	5.2115	0.1237	0.1123	0.0982
32	0.6737	14.1241	0.1756	0.1460	0.1370
33	0.7131	1.9848	0.0507	0.0399	0.0424
34	0.7092	4.0174	0.0830	0.0710	0.0734
35	0.7041	6.4394	0.1341	0.1205	0.1090
36	0.7229	7.6474	0.1971	0.1821	0.1824

Table 4A. Average number of vehicles in best solutions generated by five replications of the GA for the Burma 14 instances for four different sets of initial sequences. These replications used simple mutation and short runs.

Instance Number	Initial sequence(s) SFC	SWP	NN	SFC, SWP and NN
1	4	4	4	4
2	2	2	2	2
3	2	2	2	2
4	1	1	1	1
5	4	4	4	4
6	2	2	2	2
7	2	2	2	2
8	1	1	1	1
9	4	4	4	4
10	2	2	2	2
11	2	2	2	2
12	1	1	1	1
13	2	2	2	2
14	1	1	1	1
15	1	1	1	1
16	1	1	1	1
17	2	2	2	2
18	1	1	1	1
19	1	1	1	1
20	1	1	1	1
21	2	2	2	2
22	1	1	1	1
23	1	1	1	1
24	1	1	1	1
25	1	1	1	1
26	1	1	1	1
27	1	1	1	1
28	1	1	1	1
29	1	1	1	1
30	1	1	1	1
31	1	1	1	1
32	1	1	1	1
33	1	1	1	1
34	1	1	1	1
35	1	1	1	1
36	1	1	1	1

Table 4B. Average number of vehicles in best solutions generated by five replications of the GA for the Ulysses 22 instances for four different sets of initial sequences. These replications used simple mutation and short runs.

Instance Number	Initial sequence(s) SFC	SWP	NN	SFC, SWP and NN
1	6	6	5.8	5.6
2	3	3	3	3
3	2	2	2	2
4	2	2	2	2
5	5	5	5	5
6	3	3	3	3
7	2	2	2	2
8	2	2	2	2
9	5	5	5	5
10	3	3	3	3
11	2	2	2	2
12	2	2	2	2
13	2	2	2	2
14	1	1	1	1
15	1	1	1	1
16	1	1	1	1
17	2	2	2	2
18	1	1	1	1
19	1	1	1	1
20	1	1	1	1
21	2	2	2	2
22	1	1	1	1
23	1	1	1	1
24	1	1	1	1
25	1	1	1	1
26	1	1	1	1
27	1	1	1	1
28	1	1	1	1
29	1	1	1	1
30	1	1	1	1
31	1	1	1	1
32	1	1	1	1
33	1	1	1	1
34	1	1	1	1
35	1	1	1	1
36	1	1	1	1

Table 4C. Average number of vehicles in best solutions generated by five replications of the GA for the Berlin 52 instances for four different sets of initial sequences. These replications used simple mutation and short runs.

Instance Number	Initial sequence(s) SFC	SWP	NN	SFC, SWP and NN
1	18	18.4	18.2	18.2
2	9	9	9	9
3	5	5	5	5
4	4	4	4	4
5	16.4	16	16.2	16
6	8	8	8	8
7	4.8	4.4	4.6	4.4
8	4	4	4	4
9	15	15	15	15
10	7	7	7	7
11	4	4	4	4
12	3	3	3	3
13	6	6	6	6
14	3	3	3	3
15	2	2	2	2
16	2	2	2	2
17	5	5	5	5
18	3	3	3	3
19	2	2	2	2
20	2	2	2	2
21	5	5	5	5
22	2.8	2.8	2.8	3
23	2	2	2	2
24	1	1	1	1
25	3	3	3	3
26	2	2	2	2
27	1	1	1	1
28	1	1	1	1
29	3	3	3	3
30	2	2	2	2
31	1	1	1	1
32	1	1	1	1
33	3	3	3	3
34	2	2	2	2
35	1	1	1	1
36	1	1	1	1

Table 4D. Average number of vehicles in best solutions generated by five replications of the GA for the Bier 127 instances for four different sets of initial sequences. These replications used simple mutation and short runs.

Instance Number	Initial sequence(s) SFC	SWP	NN	SFC, SWP and NN
1	40.6	40.2	40.4	40.4
2	19	19	19	19
3	10	10.4	10	10
4	8	8.2	8	8
5	36.6	36.8	36.8	37.2
6	17.2	17	17	17
7	9.2	9.2	9	9.4
8	8	8	8	8
9	34.2	34.6	34.6	34.4
10	16	16	16	16
11	8.8	8.6	9	8.8
12	7	7	7	7
13	13	13	13	13
14	6	6.2	6	6
15	3.6	4	3	3
16	3	3	3	3
17	12	11.8	11.6	11.8
18	6	6	6	6
19	3	3	3	3
20	3	3	3	3
21	10.2	10	10	10
22	5	5	5	5
23	3	3	3	3
24	2	2	2	2
25	7	7	7	7
26	3.4	3.8	3	3
27	2	2	2	2
28	2	2	2	2
29	6	6	6	6
30	3	3	3	3
31	2	2	2	2
32	2	2	2	2
33	5.6	6	6	5.6
34	3	3	3	3
35	2	2	2	2
36	2	2	2	2

Table 5A. Average run times for five replications of the GA for the Burma 14 instances for four different sets of initial sequences. These replications used simple mutation and short runs.

Instance Number	Initial sequence(s) SFC	SWP	NN	SFC, SWP and NN
1	0.8545	0.6385	0.7632	0.7717
2	0.7435	0.8686	0.9965	0.6623
3	1.0504	0.5418	1.4002	0.5042
4	0.1208	0.1124	0.1104	0.1081
5	0.8416	0.8618	0.9500	0.8739
6	0.6362	0.7184	0.6594	0.6187
7	2.0130	1.5286	2.5972	1.4455
8	1.1084	1.1040	1.1006	1.1081
9	1.0040	0.8766	0.8369	0.8454
10	0.8447	0.8436	0.8985	0.8933
11	1.2640	1.2637	1.2623	1.2677
12	1.1642	1.1659	1.1632	1.1648
13	1.1363	1.0207	1.2322	1.1843
14	0.0999	0.1012	0.0931	0.0957
15	0.0912	0.0881	0.0864	0.0902
16	0.0919	0.0867	0.0866	0.0868
17	0.3903	0.4781	0.4904	0.4182
18	0.1047	0.0947	0.0958	0.0957
19	0.0936	0.0876	0.0884	0.0894
20	0.0902	0.0863	0.0862	0.0894
21	0.6591	0.5787	0.6379	0.7174
22	0.0993	0.0900	0.0933	0.0953
23	0.0895	0.0910	0.0868	0.0881
24	0.0921	0.0894	0.0885	0.0880
25	0.1285	0.1147	0.1126	0.1106
26	0.0953	0.0954	0.0943	0.0951
27	0.0899	0.0877	0.0882	0.0887
28	0.0888	0.0850	0.0849	0.0890
29	0.1088	0.1132	0.1098	0.1096
30	0.0989	0.0928	0.0976	0.0938
31	0.0904	0.0868	0.0864	0.0894
32	0.0875	0.0865	0.0868	0.0849
33	0.1258	0.1284	0.1247	0.1226
34	0.0922	0.0904	0.0965	0.0937
35	0.0895	0.0877	0.0875	0.0878
36	0.0868	0.0872	0.0901	0.0865

Table 5B. Average run times for five replications of the GA for the Ulysses 22 instances for four different sets of initial sequences. These replications used simple mutation and short runs.

Instance Number	Initial sequence(s) SFC	SWP	NN	SFC, SWP and NN
1	1.2328	1.2903	1.1408	0.9626
2	1.7304	1.2496	1.7154	1.2172
3	1.7266	1.6062	1.7353	2.2621
4	2.1263	1.6550	2.1955	2.4456
5	1.4316	1.5179	1.4112	1.4508
6	1.9796	1.9409	2.0659	2.6516
7	2.3248	2.0415	2.5778	1.7660
8	1.1494	2.7196	3.3119	1.4159
9	1.5858	1.3975	1.0985	1.0375
10	1.6591	2.8269	1.9410	2.4709
11	3.6161	3.3743	2.4688	2.8503
12	1.5259	1.3581	1.4424	2.0275
13	2.2569	2.0625	1.8158	2.0659
14	0.2153	0.2272	0.2103	0.2293
15	0.1959	0.1961	0.1985	0.1927
16	0.1862	0.1894	0.2018	0.1862
17	3.4121	2.5549	2.4929	2.6608
18	0.2059	0.1989	0.1979	0.2004
19	0.1998	0.1964	0.1903	0.1955
20	0.1947	0.1916	0.1859	0.1893
21	0.3417	0.3758	0.3454	0.3535
22	0.4253	0.4505	0.5502	0.4266
23	0.2009	0.1916	0.1928	0.1928
24	0.1943	0.1964	0.1939	0.1850
25	0.2577	0.2443	0.2507	0.2351
26	0.2223	0.2102	0.2116	0.2074
27	0.1870	0.1908	0.1912	0.1867
28	0.1870	0.1873	0.1920	0.1872
29	0.6577	0.7741	0.7581	0.6308
30	0.1985	0.2220	0.2139	0.1994
31	0.1899	0.1846	0.1919	0.1924
32	0.1850	0.1838	0.1840	0.1847
33	2.6973	2.7293	2.6110	2.4648
34	0.2078	0.2110	0.2090	0.2123
35	0.1959	0.1901	0.1926	0.1880
36	0.1896	0.1895	0.1828	0.1905

Table 5C. Average run times for five replications of the GA for the Berlin 52 instances for four different sets of initial sequences. These replications used simple mutation and short runs.

Instance Number	Initial sequence(s) SFC	SWP	NN	SFC, SWP and NN
1	3.8631	3.2056	3.6175	3.5456
2	2.7899	3.6435	1.6910	1.7040
3	2.5727	3.9989	4.6195	3.1642
4	5.1212	4.6721	4.6653	4.5535
5	4.2705	5.3687	3.6270	3.8427
6	4.6097	4.7565	4.9718	5.9132
7	5.1031	5.4263	4.9755	3.6282
8	5.0959	7.3038	4.1533	5.3364
9	4.4695	3.6607	7.1697	6.4802
10	4.6795	3.8261	3.5292	3.3428
11	4.7361	3.8770	7.1166	5.5300
12	4.3942	3.6415	5.2756	3.4518
13	6.3172	4.9251	5.1190	4.5433
14	7.2862	7.7693	8.0968	8.0680
15	8.5886	10.7146	9.7537	9.1530
16	13.8212	12.5638	11.3095	10.0927
17	5.4253	7.7988	3.9986	4.0217
18	6.5002	10.4881	6.1207	9.7788
19	12.0787	15.0048	13.6630	15.9585
20	7.3960	6.8192	8.7395	12.7525
21	2.9984	3.2214	2.8683	2.8771
22	4.8742	4.7718	5.1321	6.0506
23	6.6232	4.1530	5.7640	14.0572
24	13.6724	13.6827	12.9101	12.9505
25	7.2476	6.0910	6.8231	8.4757
26	12.2771	14.2342	13.9938	13.6248
27	1.0859	1.0444	1.0082	1.0178
28	1.0304	1.0333	1.0203	0.9510
29	6.7101	5.1190	5.7465	6.2653
30	8.8499	11.2243	10.0152	17.1050
31	1.1474	1.0961	1.0632	1.0376
32	1.0166	1.0207	0.9767	1.0030
33	5.5082	5.5895	5.7038	5.2740
34	8.7482	7.9959	9.7490	8.4060
35	1.1058	1.1612	1.0406	1.0665
36	0.9888	1.0153	0.9497	0.9964

Table 5D. Average run times for five replications of the GA for the Bier 127 instances for four different sets of initial sequences. These replications used simple mutation and short runs.

Instance Number	Initial sequence(s) SFC	SWP	NN	SFC, SWP and NN
1	12.2286	12.6972	12.6231	12.3973
2	11.9593	13.3357	10.9179	10.3244
3	6.6103	15.2879	13.1106	9.1703
4	14.5786	15.0912	14.0074	14.1458
5	16.1669	17.3896	11.5147	11.0007
6	11.4677	13.3766	11.3934	12.0898
7	14.6427	15.5527	14.7492	12.0596
8	14.2898	18.0836	11.3887	8.2979
9	15.1084	10.3060	15.0510	15.6147
10	9.1494	6.6239	9.8338	6.4492
11	22.9538	10.4229	25.6508	17.1166
12	31.5646	23.9675	21.3391	25.9806
13	11.2033	14.5856	18.0786	13.9414
14	23.3403	28.8014	23.3244	16.0480
15	31.2708	42.9698	20.5947	13.9106
16	30.5790	50.2740	34.5108	30.6385
17	18.9210	12.4450	10.6708	23.2236
18	21.3551	25.6860	43.9601	33.0732
19	35.2149	68.6512	39.7994	44.2835
20	31.3969	66.2408	41.0632	35.3572
21	9.3631	8.0747	8.3922	9.3085
22	22.7671	22.9663	20.7351	21.5932
23	55.6465	60.6299	54.9186	49.9905
24	33.5089	32.7125	28.7772	35.3453
25	29.2705	30.2440	33.7611	28.9785
26	51.6632	51.6230	54.2669	44.6006
27	60.0084	84.6435	51.2066	51.5031
28	84.0703	102.8928	22.6867	61.2047
29	9.9007	17.9801	25.8101	13.4411
30	76.5579	24.7793	49.9393	45.8875
31	75.1979	74.3360	90.0702	111.1157
32	92.5607	128.0737	93.6478	80.1756
33	13.2855	11.2671	12.8739	14.5679
34	56.5019	37.9880	40.7225	65.0335
35	101.2287	80.6153	79.6529	105.7196
36	76.1413	76.8837	74.7479	75.8575

Table 6. Average number of vehicles in solutions from and run times for five replications of the GA for the Berlin 52 instances using local search (with short runs) and simple mutation (with long runs).

	Local search with short runs		Simple mutation with long runs	
Instance Number	Average number of vehicles	Average run time (seconds)	Average number of vehicles	Average run time (seconds)
1	17.8	178.9564	17	357.1714
2	8.8	178.6973	9	324.0521
3	5	270.6053	5	525.2008
4	4	408.4143	4	585.1433
5	16.2	104.5844	15.8	194.8333
6	8	95.2655	8	248.1333
7	4.6	330.1949	4.6	326.3183
8	4	756.3902	4	431.9377
9	15	27.4770	15	12.6426
10	7	6.7784	7	3.5744
11	4	58.6032	4	5.5247
12	3	4.4651	3	3.0069
13	6	222.0894	6	192.1085
14	3	284.8046	3	520.2556
15	2	608.9143	2	551.6539
16	2	919.6634	2	1468.1637
17	5	18.4188	5	4.3748
18	3	150.4596	3	108.4222
19	2	745.3417	2	137.8749
20	2	894.2812	2	22.3771
21	5	33.1429	5	2.8049
22	3	16.6558	2.8	5.5593
23	2	61.6236	2	6.2294
24	1	13.6835	1	13.0081
25	3	156.9601	3	535.8973
26	2	477.7522	2	725.3166
27	1	1.1114	1	1.0560
28	1	0.9757	1	0.9723
29	3	197.1747	3	3.8092
30	2	296.2467	2	23.5356
31	1	1.0250	1	1.0030
32	1	0.9565	1	0.9599
33	3	10.3759	3	6.0284
34	2	13.8292	2	10.1665
35	1	1.0816	1	1.0822
36	1	0.9659	1	1.0218

Table 7. CRIRP problem instances and GA performance.

Instances	Initial sequence(s)			
	SFC	SWP	NN	SFC, SWP and NN
Burma 14	1.50	1.50	1.50	1.50
Ulysses 22	1.78	1.78	1.77	1.77
Berlin 52	4.31	4.29	4.30	4.29
Bier 127	9.01	9.05	8.98	8.99
Low capacity	7.85	7.85	7.85	7.84
Medium capacity	2.83	2.83	2.80	2.81
High capacity	1.77	1.79	1.77	1.76

Table 8. GA performance on the Berlin 52 instances.

GA options	Average number of vehicles	Average run time (seconds)
Simple mutation, short runs	4.29	6.2783
Local search, short runs	4.29	209.6657
Simple mutation, long runs	4.26	204.4783

8. Summary and Conclusions

This paper introduced the CRIRP, a type of inventory routing problem that has an interesting link between the elements of time and demand. The continuous replenishment means that the operating costs are related to the number of vehicles. The demand at each site is a rate (items per time unit), not a fixed amount. In the special case in which all sites have the same demand, the problem is equivalent to the bin packing problem. However, the more general case involves the traditional elements of routing as well as assignment. Experimental results show that a heuristic that finds vehicle routes with the Clarke-Wright savings algorithm creates solutions that are generally better than those generated by bin packing and route-first cluster-second heuristics. The genetic algorithm did not produce better solutions, even when local search was used or when replications were allowed to run much longer.

This work has focused on formulating the problem and suggesting some approaches that can generate high-quality solutions quickly. Additional work will consider improving the lower bound, refining the heuristics (by using different route construction heuristics or including route improvement techniques, for instance), and developing exact methods such as column generation and branch-and-cut procedures.

Acknowledgements

Kay Aaby, Rachel Abbey, Carol Jordan, and Kathy Wood at the Montgomery County, Maryland, Public Health Services provided helpful information about the medication distribution problem. Cooperative Agreement Number U50/CCU302718 from the CDC to NACCHO supported this publication. Its contents are solely the responsibility of the University of Maryland and the Advanced Practice Center for Public Health Emergency Preparedness and Response of Montgomery County, Maryland, and do not necessarily represent the official views of CDC or NACCHO.

References Cited

Bard, Jonathan F., Liu Huang, Patrick Jaillet, and Moshe Dror, "A decomposition approach to the inventory routing problem with satellite facilities," *Transportation Science*, Volume 32, Number 2, pages 189 – 203, 1998.

Bartholdi, John J., III and Loren K. Platzman, "Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space," *Management Science*, Volume 34, Number 3, pages 291-305, 1988.

Beasley, JE, "Route First-Cluster Second Methods for Vehicle Routing," *OMEGA*, Volume 11, Number 4, pages 403-408, 1983.

Campbell, A., Clarke, L., Kleywegt, A., and Savelsbergh, M., "The Inventory Routing Problem," *Fleet Management and Logistics* (eds. Crainic, G. L. and Laporte, G.), Kluwer Academic Publishers, Norwell, Massachusetts, pages 95-113, 1998.

Campbell, Ann Melissa, and Martin W. P. Savelsbergh, "A Decomposition Approach for the Inventory-Routing Problem," *Transportation Science*, Volume 38, Number 4, pages 488–502, November 2004a.

Campbell, Ann Melissa, and Martin W. P. Savelsbergh, "Delivery Volume Optimization," *Transportation Science*, Volume 38, Number 2, pages 210-223, May 2004b.

Clarke, G., and J.W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, Volume 12, Number 4, pages 568-581, 1964.

Fomundam, Samuel, *Solving Continuous Replenishment Inventory Routing Problems*, M.S. Thesis, University of Maryland, College Park, 2008.

Gillett, Billy E., and Leland R. Miller, "A Heuristic Algorithm for the Vehicle-Dispatch Problem," *Operations Research*, Volume 22, Number 2, pages 340-349, 1974.

Golden, B., L. Bodin, T. Doyle, and W. Stewart Jr., "Approximate Traveling Salesman Algorithms," *Operations Research*, Vol. 28, No. 3, Part 2, pp. 694-711, 1980.

Golden, Bruce, Arjang Assad, and Roy Dahl, "Analysis of a large scale vehicle routing problem with an inventory component," *Large Scale Systems*, Volume 7, pages 181-190, 1984.

Jaillet, Patrick, Jonathan F. Bard, Liu Huang, and Moshe Dror, "Delivery cost approximations for inventory routing problems in a rolling horizon framework," *Transportation Science*, Volume 36, Number 3, pages 292-300, 2002.

Lenstra, J.K., and A. Rinnooy Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, Volume 11, pages 221-227, 1981.

Moin, NH, and S Salhi, "Inventory routing problems: a logistical overview," *Journal of the Operational Research Society*, Volume 58, pages 1185–1194, 2007.

Prins, C, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, Volume 31, pages 1985-2002, 2004.

Toth, P. and Vigo, D. 1998. "Exact Solution of the Vehicle Routing Problem," *Fleet Management and Logistics* (eds. Crainic, G. L. and Laporte, G.), Kluwer Academic Publishers, Norwell, Massachusetts, 1-31.

TSPLIB, http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html, accessed July 8, 2008.

Webb, Ian R., and Richard C. Larson, "Period and phase of customer replenishment: a new approach to the strategic inventory/routing problem," *European Journal of Operational Research*, Volume 85, pages 132-148, 1995.